(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 152 352 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
07.11.2001 Bulletin 2001/45

(51) Int Cl.[7]: **G06F 17/30**

(21) Application number: **01000121.2**

(22) Date of filing: **20.04.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **27.04.2000 US 561252**

(71) Applicant: **International Business Machines Corporation**
Armonk, NY 10504 (US)

(72) Inventors:
- **Cannon, David M.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Dawson, Colin S.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Haye, Mark A.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Smith, James P.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)

(74) Representative: **Burt, Roger James, Dr.**
IBM United Kingdom Limited,
Intellectual Property Law,
MP 110,
Hursley Park,
Hursley
Winchester, Hampshire SO21 2JN (GB)

(54) System and method for handling files in a distributed data storage environment

(57) A system and method for relating files in a distributed data storage environment is described that allows for positive identification of membership of a file within a group, even in a loosely coupled environment where files are not available for comparison in real time. In disclosed embodiments, base files of a client are stored on a server and are accompanied by tokens uniquely identifying the base files. The tokens are generated on the client and may be derived from the contents of the base file using a digital signature. Each file transmitted to the server is accompanied with a token. Incremental backups may be used, and may employ file differencing. Accordingly, sub-files related to the base files may be transmitted to the server for backup. The sub-files are related to their respective base files using the tokens and are cross-linked to the base files so that any sub-files can be retrieved together with the base file from which the sub-file was derived.

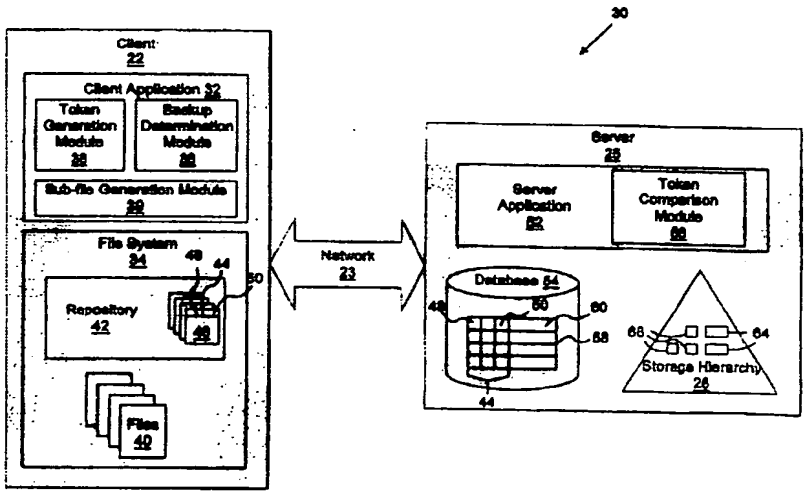


Fig. 3

EP 1 152 352 A2

Description

[0001] The present invention relates to systems and methods for handling files in a distributed data storage environment.

[0002] In a data processing system, a backup/restore subsystem, usually referred to as a backup subsystem, is typically used as a means to save a recent copy or version of a file, plus some number of earlier versions of the same file, on some form of backup storage device such as a magnetic disk drive, tape, or optical storage device. The backup subsystem is used as a means of protecting against loss of data in a given data processing system. For example, if an on-line version of a file is destroyed or corrupted because of power failure, hardware or software error, user error, or some other type of problem, the latest version of that file which is stored in a backup subsystem can be restored and therefore the risk of loss of data is minimized. Another important use of backup subsystems is that even if failures do not occur, but files or data are deleted or changed (either accidentally or intentionally), those files or data can be restored to their earlier state thus minimizing the loss of data.

[0003] A closely related concept to the backup subsystem is an archive/retrieve system, usually referred to as an archive subsystem. Archiving refers to making copies of files on lower cost storage such as tape so that the files can be deleted from more expensive technology such as disk storage. Since disk storage is frequently being updated, an archival copy also helps in preserving the state of a collection of data at a particular point in time. Although the improved method of carrying out the backup disclosed in this application is primarily described for a backup system, it will be apparent to the person of ordinary skill in the art of data processing that the systems and methods described herein are also applicable to archive systems or other related data storage and storage management systems.

[0004] At the present time, the majority of backup systems run on host systems located in a data processing environment. Typically, a new version (also referred to as changed version) of a file is backed up based on a predetermined schedule, such as at the end of each day, or after each time that a file has been updated and saved.

[0005] Backup systems generally consume large amounts of storage media, because multiple versions of large amounts of data are being backed up on a regular basis. The transmission of the large amounts of data that prior art backup systems necessarily store also consumes large amounts of network bandwidth. Therefore, those engaged in the field of data processing and especially in the field of backup/restore systems are continuously striving to find improved methods and systems to reduce the storage demand in backup systems. Previously, a full backup was conducted for each file in a system. More recently, an incremental backup method has been employed to enable the storage of and retrieval of multiple versions of a given file while consuming less storage space.

[0006] The full backup method is the most basic method used and requires the back up of an entire collection of files, or a file system, regardless of whether individual files in that collection have been updated or not. Furthermore, in the full backup method, multiple full versions of each file are maintained on a storage device. Since maintaining multiple full copies of many files consumes substantial amount of storage, compression techniques are sometimes used to reduce the amount of data stored. Compression techniques basically rely on the presence of redundancy within the file. so called intra-file redundancy, in order to achieve this reduction. The most common method is the use of a method of file compression known as Lempel-Ziv method (also known as Adaptive Dictionary Encoder or LZ coding) described in a book by T. C. Bell et. al, titled Text Compression, pp 206-235. The essence of Lempel-Ziv coding is that redundant phrases are replaced with an alias, thereby saving the storage space associated with multiple occurrences of any given phrase. This is a general method which can be applied to any file and typically results in compression ratios of the order of between 2 and 3.

[0007] Incremental backup is an alternative to full backup. In systems using incremental backup, backups are performed only for those files which have been modified since the previous incremental or full backup.

[0008] In any given backup system, the higher the backup frequency, the more accurately the backup copy will represent the present state of data within a file. Considering the large volume of data maintained and continuously generated in a typical data processing system, the amount of storage, time, and other resources associated with backing up data are very substantial.

[0009] Aside from the compression technique which is heavily utilized to reduce storage requirement in a backup system, there exists a different method of achieving a reduction in backup file size. This method is known as delta versioning or "differencing."

[0010] Differencing relies on comparisons between two versions of the same file, where multiple versions are saved as a "base file," together with some number of "sub-files" which represent only the changes to the base file. These small files, also referred to as "delta files" or "difference files," contain only the changed portions, typically bytes or blocks which have changed from the base file. Delta files are generated as a result of comparing the current version of a file with an earlier version of the same file, referred to as the base file. Differencing thus exploits redundancy between file versions, in order to achieve reductions in storage space and network traffic.

[0011] Substantial storage savings in backup systems may result from the adoption of differencing techniques, since frequently the selection of a file for incremental backup occurs after a small change has been

made to that file. Therefore, since many versions of a file that differ only slightly from one another may be backed up, differencing offers great potential for substantial reductions in the amount of data that must be transferred to and stored in the backup server.

[0012] Recently, the emergence of low cost local area networking, personal computer, and workstation technology has promoted a new type of data processing architecture known as the "client-server" system or environment. A client-server system 10, as shown in FIG. 1, typically consists of a plurality of client computers (also referred to as clients) 11, such as personal computers or workstations. The client computers 11 are preferably provided with a local storage medium 12 such as a disk storage device. The client computers 11 communicate over a network 13, such as an Ethernet or a Token Ring, which links the clients 11 to one or more network server computers 14.

[0013] The server computer 14 is generally a mainframe computer, a workstation, or other high end computer and is typically provided with one or more local storage mediums 15 such as a disk storage device, a tape storage device, and/or an optical storage device. The server computer 14 usually contains various programs or data which are shared by or otherwise accessible to the clients 12. Such a client-server system communicating over a network is often referred to as a "distributed" system or network.

[0014] The distributed client-server environment presents a number of major issues related to data processing, integrity, and the backup of such data. One major concern in the client-server environment is that a substantial amount of critical data may be located on client subsystems which lack the security, reliability or care of administration that is typically applied to server computers. A further concern is that data may accidentally be lost from a client computer, as users of such computers often do not take the time and care necessary to back up the data on a regular basis. Another concern is that backing up large amounts of data from a client can require large amounts of network bandwidth and server storage space.

[0015] Recently a number of client-server backup systems have been developed to alleviate some of the concerns listed above. An example is IBM's Tivoli Storage Manager (TSM), formerly known as ADSM (ADSTAR Distributed Storage Manager). This technology overcomes some of the deficiencies mentioned above by making backup copies of the client data on a backup server. The client copies are made automatically without user involvement and are stored on storage devices which are administered by the backup server.

[0016] A typical client-server backup system such as TSM typically operates with a client application operating in the client computer 11 and a server application operating in the server computer 14. The client application, also known as a client backup program, is activated at pre-specified or periodic times and makes contact with the server application, also referred to as a server backup program. After establishing contact and performing authentication, the client application then consults a user-configurable policy which instructs the client application regarding which sort of a backup operation should occur and which files on the client computer will be the subjects of the current backup. It then searches all or a subset of files on the client computer, determining which files should be backed up.

[0017] For example, a data file which has changed since the last backup was conducted may be selected for the backup operation. After selecting the files to be backed up, the client application transmits those files across the network to the server application. The server application then makes an entry in a listing such as a backup catalog for each file received and stores those files on storage devices attached to the backup server.

[0018] The backup system, in order to efficiently manage data storage may store data in storage devices organized in a storage hierarchy. A storage hierarchy provides a number of levels of storage device with data storage in devices at the top levels being more expensive but having shorter access times. Moving down the hierarchy, data storage becomes less expensive, but the access times are longer. Accordingly, frequently accessed data is stored at the higher levels, while the lower levels are more suitable for long-term data storage. Among the levels of the hierarchy, data is stored in storage pools. A storage pool is a collection of storage volumes with similar geometries. Pools are collections of volumes capable of being used on a particular device. Examples of media stored in pools include tape, optical disks, magnetic disks, and other media having the same format.

[0019] The backup system also carries out several other important operations. For instance, backup copies of files that were made many months ago may be moved from disk storage to tape storage in order to reduce storage costs. Another important function of the client-server backup system occurs when the user requests the restoration of a file. The client application contacts the server application, which consults its backup catalog to establish the location of the backup copy of the file. It then returns that file across the network to the client computer which in turn makes it available to the user.

[0020] Examples of hardware which may be employed in a backup system in a distributed client-server environment include one or more server computers such as mainframes, workstations, and other high end computers and storage mediums such as the IBM 3390 magnetic storage system, IBM 3494 tape storage library or IBM 3595 optical library. Optical and tape storage libraries typically provide automated mechanical mounting and demounting of tape or optical cartridges into read/write drives. When several such devices are present, the server application is often configured to utilize the devices in a storage hierarchy in which the most likely to be accessed backup files are kept on faster access devices such as local non-volatile memory, and

files less likely to be accessed are kept on less expensive, but slower access devices, such as tape or optical disks.

[0021] Despite the recent improvements made in the field of distributed client-server backup systems, certain shortcomings remain in currently available systems. Primary among these shortcomings is that the very large amounts of data on the clients now being regularly backed up tend to require large amounts of network bandwidth and to require high quantities of server storage space, which can be quite costly. Although storage management systems such as TSM may compress this data on the storage devices, the amount of data remains very large. Differencing is thought to be a solution to this problem, but differencing poses certain problems in itself.

[0022] For instance, in a differencing backup system, once a base file is stored in the storage devices, the base file may not be available for immediate inspection. Often, the backup server is configured with a plurality of storage devices, such as optical devices, tape backups, and non-volatile memory (such as hard disk drives) organized in the above-described storage hierarchy. Within the storage hierarchy, the particular optical disks or tapes are frequently swapped out, and the only copy of a base file may be on a disk or tape that is not currently mounted. In addition, even when the base files are immediately available on such devices, accessing the base files and scanning the devices for the base files is a relatively slow process.

[0023] Current backup systems using the differencing method of backup typically store information about the files previously backed up to the server. This information helps determine the current state of backed up files and whether these files are still available. Nevertheless, for one reason or another, the versions of the backed up files may have changed between the client and the server. For instance, either the client's record of the files or the server's version of the files may have been deleted or inadvertently altered.

[0024] Accordingly, when a sub-file is transferred to the server, a reliable method is necessary to identify or "relate" the sub-file with the base file from which it was derived in order to later be able to combine the sub-file with its base file during a restore operation. If a sub-file is not restored with the correct corresponding base file, it is not possible to correctly reconstruct the original file, and a data integrity error occurs.

[0025] Certain additional challenges exist in relating sub-files to base files in a distributed environment. These stem from the fact that the elapsed time between backups of the base file and a dependent sub-file could be highly variable. Additionally, the client's record of base file information could be invalid. For instance, the sub-file backup algorithm may have been disabled either on the client or on the server. Additionally, a client may back up data to multiple servers, causing the client's knowledge of the base file to be invalid relative to one or more of the different servers. Furthermore, the server database may have been regressed to an earlier point in time in the interim between storing the base file and generating a sub-file. This might occur, for instance, as a result of the database becoming corrupted and being restored from an older version. Accordingly, as discussed, the base files the server knows about may not match those which the client has tracked. It is therefore apparent that implementation of an efficient backup subsystem in a computer processing environment is a formidable task and implementing such a system in a distributed client-server environment poses significant challenges.

[0026] Accordingly, the present invention provides a system, method and computer program as described in the appended claims, whereby groups of files can be transmitted to a remote storage site using an identifier unique to each group. The approach adopted permits an improved backup system and method in a client-server environment that not only substantially reduces the storage and network bandwidth requirements of current backup systems, but also minimizes the burden in communicating the relationships between groups of files, such as base files and their sub-files, between a client and a server. The integrity of the system is maintained through positive identification of the relationships between groups of files transmitted between the client and server.

[0027] In a preferred embodiment of the invention the data storage management system relates groups of files in a distributed data storage management system having a primary storage site such as a client computer and a remote storage site such as a server computer. The primary storage site and the remote storage site communicate over a network. A token generation module is located within the primary storage site and configured to generate tokens uniquely identifying groups of files of the primary storage site. A token listing is readily available within the remote storage site and a token comparison module is also located within the remote storage site. The token comparison module is preferably configured to receive tokens passed together with a file from the primary storage site to the remote storage site and compare the tokens to one or more tokens within the token listing to establish the relationship (if any) of the file with other files previously transmitted from the primary storage site to the remote storage site.

[0028] Typically a plurality of base files are resident within the storage devices of the remote storage site and a unique token for each of the base files is stored within the token listing. A plurality of tokens may be stored within the token listing, and each of the plurality of tokens uniquely identifies a base file resident within the storage devices of the remote storage site.

[0029] A backup determination module is preferably resident within the primary storage site and is configured to select files for storage on the remote storage site, and determine whether the files should be stored as base

files or sub-files. If the files are to be stored as sub-files, a sub-file generation module generates the sub-files by comparing the current file with a previously backed up base file. Thus, a plurality of sub-files can also be stored within the storage devices of the remote storage site, and each of the plurality of sub-files is preferably cross-linked with a base file resident within the storage devices.

[0030] The preferred embodiment also comprises a repository located within the primary storage site. The repository preferably contains a representation of each of a plurality of base files stored on the remote storage site and also preferably stores a token unique to each of the base files together with the base files. A token generation module is configured to generate tokens at least partially indicative of the contents of the base files and may be configured to generate tokens comprising two components, a file identifier comprising attributes of a base file and an identification key derived from the contents of a base file.

[0031] The preferred embodiment of the invention also encompasses a data storage management method for relating groups of files in a distributed data storage management system. This involves assigning a token to a base file of the primary storage site. The token uniquely identifies the base file and may be comprised of two components, a file identifier comprising attributes of the base file and an identification key derived from the contents of the base file. A copy of the base file is then passed from the primary storage site to the remote storage site, where the base file is stored on a storage medium of the remote storage site. A copy of the token assigned to the base file is passed together with the base file from the primary storage site to the remote storage site. The token is then stored in a token listing of the remote storage site.

[0032] In accordance with the preferred embodiment, a sub-file may be derived from the base file and the current file. A second token copied from or based upon the token of the base file is then associated with the sub-file and passed together with the sub-file to the remote storage site. The remote storage site relates the sub-file to the base file by comparing the second token to the token listing and matching the token of the base file. Thereafter, a cross-linking between the sub-file and the base file is generated, and the sub-file is stored in the storage hierarchy. Consequently, in response to a restore request from the primary storage site, the sub-file and the base file can be returned together to the primary storage site from the remote storage site.

[0033] Viewed from another perspective the present invention provides a method for relating groups of files in a distributed data storage system having a primary storage site and a remote storage site, the method comprising: assigning a token to a base file of the primary storage site, the token uniquely identifying the base file and comprised of two components, a file identifier comprising attributes of the base file and an identification key derived from the contents of the base file; passing a copy of the base file from the primary storage site to the remote storage site; transferring the base file to a storage medium attached to the remote storage site; passing a copy of the token from the primary storage site to the remote storage site; storing the token in a token listing of the remote storage site; deriving a sub-file from the base file, assigning a second token based upon the token of the base file to the sub-file, and passing the sub-file together with the second token to the remote storage site; determining at the remote storage site the relation of the sub-file to the base file by comparing the second token to the token listing and matching the token of the base file; creating a cross-linking between the sub-file and the base file; and in response to a request from the primary storage site, returning the sub-file and the base file substantially together to the primary storage site from the remote storage site.

[0034] The present invention also provides a computer program for implementing the methods described above. Such a computer program typically comprises machine instructions for causing a system (or multiple systems if distribution across machines) to perform the relevant method steps. The computer program instructions can be supplied on a suitable storage medium, such as a CD ROM, or provided for download over a network. It will further be appreciated that the computer program, method and system of the invention will all generally benefit from the same preferred features.

[0035] A preferred embodiment of the invention will now be described in detail by way of example only with reference to the following drawings:

Figure 1 is a schematic block diagram illustrating one embodiment of a typical distributed client-server system of the prior art.

Figure 2 is a schematic block diagram illustrating one embodiment of a distributed client-server system having a backup system in which the method of the present invention may be implemented.

Figure 3 is a schematic block diagram illustrating a system for relating groups of files in a distributed environment in a preferred embodiment of the present invention.

Figure 4 is a schematic block diagram illustrating a listing of grouped files in accordance with a preferred embodiment of the present invention.

Figure 5 is a schematic flow chart diagram illustrating a client-side method in accordance with a preferred embodiment of the present invention for relating groups of files in a distributed data management system.

Figure 6 is a schematic flow chart diagram illustrat-

ing a server-side method in accordance with a preferred embodiment of the present invention for relating groups of files in a distributed data management system.

[0036] Referring now to Figure 2, shown therein is a distributed client-server system 20. The system 20 typically includes a plurality of client computers 21, each with its own local storage medium 22, such as a disk storage device. The client computers (clients) 21 may typically be personal computers of the type having a system unit (not shown) which includes a CPU (processor), I/O control, and semiconductor and magnetic memories and a Windows or Macintosh operating system. The client computers 21 may further be workstations of the type having AIX, UNIX, or equivalent operating systems. These operating systems are well known to those skilled in the art of computer systems.

[0037] The client-server system 20 further includes a network 23 such as Ethernet or Token Ring which provides the communication link between the clients 21 and the backup server 25. The backup server 25 may be an IBM PC-compatible computer of the type having a system unit (not shown) which includes a CPU (processor), I/O control, and semiconductor and magnetic memories and Windows operating system. It may also be a workstation having a system unit and UNIX or AIX or equivalent operating system. It may also be a large system running the AS/400, VM or MVS operating system. The backup server 25 is also shown provided with a storage hierarchy of attached storage mediums 26 such as a disk storage device 27, optical library storage device 28, and/or tape library storage device 29.

[0038] In a client-server system 20 such as that shown in Figure 2, the backup system may reside at the backup server 25 and also have a client-side program (client application) as described above. Examples of a typical backup system distributed over a client and server include IBM's Tivoli Storage Manager (TSM), the basic operation of which has been briefly described in the background section. The operation and physical implementation of personal computers, workstations, disk storage devices, optical libraries, tape libraries, and their constituents are well known to those skilled in the art of data processing and require no further description.

[0039] Figure 3 shows a system 30 in accordance with a preferred embodiment of the present invention for relating groups of files in a distributed data storage environment. Shown within the system 30 is a primary storage site, such as a client 22 of Figure 2, and a remote storage site, such as the backup server 25 of Figure 2. The client 22 and the server 25 are in communication over the network 23 such as that of Figure 2.

[0040] Within the client 22 is shown a client application 32 and a file system 34. The client application 32 is, in one embodiment, the IBM Tivoli Storage Manager (TSM) or a similar storage management program. In the depicted embodiment, the client application 32 is provided with a backup determination module 36, a token generation module 38, and a sub-file generation module 39. The file system 34 is preferably located within non-volatile memory of the client 22, and in the depicted embodiment comprises a plurality of client files 40 and a repository 42. The client files 40 may be any type of digital data, including application files, data files, data within a database, and the like.

[0041] The backup determination module 36 is preferably programmed or otherwise configured to determine which of the files 40 have not been backed up recently and to schedule those files 40 to be backed up. Those backups are preferably conducted in an incremental manner as described above and more preferably are conducted using differencing. Accordingly, the backup determination module 36 is preferably configured to determine whether an entire backup of a file 40 must be conducted or whether only a portion of the file 40 need be backed up. When the entire file is backed up, it is stored on the server 25 as a base file 64. If a base file 64 has previously been stored for a given file 40, the backup determination module 36 may decide to store only a portion of the file 40, including one or more bytes or blocks, as a sub-file 66. The generation of the sub-file 66 is, in one embodiment, conducted by the sub-file generation module 39.

[0042] The token generation module 38 is preferably configured to generate a token 44 when the base file 64 is first transmitted to the server. Each token 44 is preferably stored within the repository 42 together with a representation 46 of the particular base file 64 which the token 44 represents. The representation 46 may be an entire copy of the base file 64 if the base file 64 is small, or may be a compressed version of the base file 64 if the base file 64 is large. The token is, in one embodiment, generated with two components, attributes (or metadata) 48 of the base file 64 and a key 50 uniquely identifying the base file 64.

[0043] In one embodiment, the key 50 is at least partially representative of the contents of the base file 64 and may be derived from the contents of the base file 64. For instance, the key 50 may be a time stamp or a compressed version of the base file 64, such as a digital signature. Preferred manners of generating digital signatures include hashing, Cyclical Redundancy Check (CRC) encoding, and checksum generation.

[0044] Shown provided within the server 25 are a backup server application 52, a database 54, a token comparison module 56, and the storage hierarchy 26 of Figure 2. The server application 52 is preferably the counterpart to the client application 32, and as such, may be part of a storage management system such as IBM's TSM. The server application 52 is preferably programmed or otherwise configured to receive the base files 64 transmitted from the client 22 and store the base files 64 within the storage hierarchy 26. In so doing, the base files 64 may become unavailable or impractical to access for comparison when sub-files 66 derived from

or otherwise related to (grouped with) the base files 64 are received. Accordingly, in order to establish membership in a common group, including the relation of a sub-file 66 to a parent base file 64, the tokens of the base files are stored within a token listing. In the depicted embodiment, the token listing is a table 58 of the database 54.

[0045] The database table 58 is provided with a series of fields, one of which contains the tokens 44 therein. The attributes 48 and key 50 of each token 44 may be stored in fields, and additional fields 60 may be included indicating the location of the base file 64 within the storage hierarchy 26. In addition, the database table 58 is preferably also used to store information about the sub-files 66 in order to provide the capability of locating the sub-files and matching the sub-files 66 to the base files 64. This information may include the location of each sub-file 66 within the storage hierarchy 26 as well as an identification code or pointer cross-linking the sub-files 66 to their respective base files 64.

[0046] The token comparison module 56 compares the tokens of each sub-file received in the server 25 against the tokens 44 within the database table 58. A match between a token transmitted with a sub-file 66 and a token of the table 58 representing a base file 64 establishes the membership of the particular sub-file 66 in the group of which the base file 64 is the primary member. In one embodiment, the group comprises the base file 64 from which the sub-file 66 was derived as well as any other sub-files derived from the base file 64. Once membership is established, the sub-file 66 may be cross-linked to the base file 64 in order to track the files when stored within the storage hierarchy 26 and for later retrieval of members of the group.

[0047] For instance, it may be desired to restore a file 40 of the client 32 to a particular point in time. The attributes 48 may be used to establish and locate the particular version of the file that was current for the desired time, whether a base file 64 or a sub-file 66. If the version is a sub-file 66, the sub-file 66 is accessed, and the cross-linking is utilised to locate the base file 64. The two files 64, 66 are then returned together to the client application 32, which uses the sub-file 66 and the base file 64 to restore the desired version of the file 40. If the version is a base file 64, the base file 64 is returned.

[0048] Figure 4 depicts a portion of a database table 58 listing. Listed therein are groups of files consisting of base files 64 and sub-files 66. Each of the base files 64 and the sub-files 66 corresponds to a separate backup of a common file 40 of Figure 3. The first listed backup occurs as base file 64c and occurred at 10:00 on 1/2/2000. The second listed backup transpired after relatively slight changes and is backed up as a sub-file 66e, occurring at 12:00 on 1/2/2000. The third backup was also a relatively slight change and accordingly was backed up as a sub-file 66f at 14:00 on 1/2/2000.

[0049] Thereafter, significant changes to the file 40 were made, and accordingly, the file 40 was backed up as a base file 64 at 15:00 on 1/3/2000. The next backup was made as a sub-file 66d at 16:00 on 1/3/2000. Thereafter, once again, significant changes were made, and the file 40 was backed up as a base file 64a, after which backups were made as sub-files 66a, 66b, and 66c.

[0050] As depicted in Figure 4, the attributes may not uniquely identify each base file 64, or relate the sub-files 66 of a base file 64 to the base file 64. Accordingly, the tokens 44, and particularly, the keys 50 are shown uniquely identifying each base file 64 and enabling reliable association of the groups of base files 64 and sub-files 66 at the server 25. A token (key) corresponding to the base file from which a particular sub-file 66 was derived is also shown associated with each sub-file 66 in order for the token comparison module to relate and group the sub-files 66 with the base files 64 when received and then cross-link the groups of base files 64 and sub-files 66.

[0051] The base files 64 and sub-files 66 may not be associated together in the storage hierarchy 26 as they are received at different times, and indeed, may be stored on different storage devices, disks, tapes, recordings, etc.

[0052] Figure 5 is a schematic flow chart diagram illustrating one embodiment of a client portion of a method 70 of relating groups of files in a distributed environment. The method 70 is preferably implemented with the system 30 of Figure 3. The method 70 starts at a step 72 and proceeds to a step 74 in which the backup determination module 36 of Figure 3 determines whether a backup of one or more of the files 40 needs to be conducted. This determination may utilise the policy discussed above, and may be based on time, on storage space constraints, upon notification of a change to a file 40, or may be initiated manually by a user.

[0053] At a step 76, the file or files 40 to be backed up are identified. At a decision step 78, the method 70 determines whether the file 40 to be backed up will be backed up as a base file 64 or as a sub-file 66. This determination may hinge on whether the file 40 has been backed up before, on how much of the file 40 has changed since the last backup, and/or whether a current representation of the base file 64 exists within the repository 42. For instance, if the file 40 has never been backed up, it is preferably initially backed up as a base file 64. If a base file 64 backing up the file 40 has already been saved to the backup server 25 and the changes are not extensive, the particular bits or blocks of data affected by the changes may be backed up as a sub-file 66. If entries for a base file 64 already exist in the repository 42 and the changes are extensive, the backup evaluation module may choose to backup the entire file as a base file 64, possibly overwriting entries in the repository 42 for the previous base file 64 corresponding to the file 40.

[0054] If the file 40 is to be backed up with a base file 64 at the current time, the method 70 proceeds to a step 80. At the step 80, a representation 46 of the base file

64 is stored within the repository 42. The representation 46, as discussed above, may be a copy of the base file 64 (which is likewise a copy of the file 40) or may be a compressed version of the base file 64. The representation 46 is used later by the backup determination module 36 to determine which changes have been made between the current file 40 and the previously backed up base file 64, as indicated by the representation 46 in the repository 42. At a step 82, a token 44 is generated for the base file 64. As discussed, the token preferably uniquely identifies the base file 64 and may be a digital signature or other representation of the contents of the base file 64.

[0055] At a step 84, the token 44 is stored within the repository 42 and is preferably linked with the representation 46 of the base file 64. At a step 86, the token 44 is transmitted to the server 25. At a step 88, the base file 64 is transmitted to the server 25, preferably in the same transaction as the transmission of the token 44 of step 86. The method 70 then progresses to a step 94, where the client-side portion of the method 70 ends. The method 70 then progresses to the node 102 of Figure 6.

[0056] Regressing back to the decision step 78, if the file 40 is to be backed up as a sub-file 66, the method 70 progresses to a step 90 where the sub-file 44 is generated. As mentioned, this preferably comprises the sub-file generation module 39 of Figure 3 comparing the current version of the file 40 to be backed up with its last backed up state, as determined by the representation 46, and placing the changed portions into the sub-file 66.

[0057] At a step 91, the token 44 of the base file 64 from which the sub-file 66 was derived, or a derivation or representation of that token is assigned to the sub-file 66. The token 44 is then transmitted to the server 25 at a step 92, preferably substantially together with the transmission of the sub-file 66 as indicated by a step 93. It is preferred that the sub-file 66 and the token 44 be transmitted together within a single transaction such that the association between the sub-file 66 and the token 44 is not lost. The method 70 then progresses to the step 94 where the client-side portion of the method 70 ends.

[0058] Figure 6 is a schematic block diagram illustrating a server-side portion of the method 70 for relating groups of files in a distributed environment in accordance with a preferred embodiment of the present invention. The server-side portion of the method 70 starts at a step 102 and progresses to a step 104. At the step 104, the token 44 transmitted at either step 88 or step 93 of Figure 5 is received by the server 25. Preferably, the token 44 is received into the server application 52 for examination of the token 44.

[0059] At a step 106, the server 25 receives the file, transmitted at step 86 or 92, that is associated with the token 44. The file is preferably either a base file 64 or a sub-file 66, but may be any file associated with a group of files. Preferably, the file is transmitted over the network 23 from the client 22 to the server 25, and as discussed, is preferably transmitted in the same transaction as the token 44. At a decision step 108, the server application 52 determines whether the file is a base file 64 or a sub-file 66. Preferably, a portion of the attributes 48 of the token 44 lists the nature of the file, whether it is a base file 64 or a sub-file 66.

[0060] If the file is determined to be a base file 64, the method 100 progresses to a step 110 where the token 44 transmitted with the base file 64 and generated at the step 82 of Figure 5 is stored within the token listing. As discussed, the token listing is preferably a table 58 of the database 54. Preferably, the location to which the base file 64 is to be stored is also stored within the database 54 together with the token 44 containing the attributes 48 and the unique key 50.

[0061] At a step 112, the base file 64 is stored within the storage hierarchy 26 of Figure 3. As previously noted, the base file 64 may be stored in a storage device connected to the server 25, such that the base file 112 is not readily available for comparison when sub-files 66 derived from the base file 64 are subsequently transmitted. Accordingly, when such sub-files are transmitted, as determined by the decision step 108, the method 100 progresses to a step 114 where the token 44 accompanying the transmitted file is compared to the tokens 44 within the token listing (the database table 58). Comparing the tokens 44 eliminates the need to have the base file 64 readily available and the unique identification key 50 of the token 44 allows for reliable identification of the base file 64 to which the sub-file corresponds.

[0062] As indicated by a decision step 115, the token comparison module 56 determines whether a base file 64 with a token 44 corresponding to the token 44 of the transmitted sub-file 66 file is listed within the table 58. If a corresponding base file 64 is not present, the method 70 progresses to a step 116. At the step 116, an error message is sent back to the client 22. The error message is a signal to the client application 32 that the file 40 being backed up should be backed up as a base file 64 rather than as sub-file 66, because no corresponding base-file 64 can be located. The method 70 then progresses to a step 117 where the received file is discarded and the server-side method then awaits the client 22 to transmit a backup of the original file 40 as a base file 64. The client 22 then preferably retransmits the backup of the file 40 as a base file, returning the method 70 to step 80 of Figure 5.

[0063] When the token comparison module 56 does locate a corresponding base file 64 at the decision step 115, the method 70 proceeds to a step 118. At the step 118, the token comparison establishes a group to which the sub-file 66 belongs. In the depicted embodiment, the group corresponds to the base file 64 and any related sub-files 66 that are also members of the group. At a step 119, the sub-file 66 is cross-linked to the base file 64. This preferably corresponds to listings within the table 58 of the locations of the base file 64 and the sub-file and an association of the base file 64 and the sub-

file 66 such that the two can be accessed together when requested by the client 22.

[0064] At a step 120, the sub-file 66 is stored within the storage hierarchy 26. Preferably, as discussed, the storage location of the sub-file is correspondingly stored within the database 54. At a step 122, the method 100 ends.

[0065] Although the preferred embodiment has been described with respect to one example where the groups of files to be related comprise base files and sub-files in a data storage management system, one skilled in the art will readily recognise that the invention has broader application and is also useful for relating other types of groups of files shared between a first storage site and a second storage site.

Claims

1. A method for relating groups of files in a distributed data storage system having a primary storage site and a remote storage site, the method comprising:

   assigning a token to a file of the primary storage site;
   passing a copy of the file from the primary storage site to the remote storage site;
   passing a copy of the token from the primary storage site to the remote storage site; and
   determining membership of the file within one of a plurality of groups residing on the remote storage site by comparing the token with other tokens on the remote storage site.

2. The method of claim 1, wherein a group of files comprises a base file and one or more sub-files derived from the base file, and said file being passed from the primary to the remote storage site may be a base file or a sub-file.

3. The method of claim 2 , wherein the token uniquely identifies the base file.

4. The method of claim 3, wherein the token is generated from the contents of the base file.

5. The method of claim 4, wherein the token is comprised of two components, a file identifier comprising attributes of the base file and an identification key derived from the contents of the base file.

6. The method of any of claims 2 to 5, wherein the file being passed from the primary to the remote storage site comprises a sub-file derived from a base file, and the token is at least partially derived from the base file.

7. The method of claim 6, further comprising assigning a second token based upon the token of the base file to the sub-file, and passing the sub-file together with the second token to the remote storage site.

8. The method of any of claims 2 to 7, further comprising, in response to a request from the primary storage site, returning a sub-file and corresponding base file to the primary storage site from the remote storage site, the relationship of the base file and the sub-file having been established with the use of the token.

9. The method of any of claims 2 to 8, wherein determining membership comprises comparing the token to a listing of tokens, each token uniquely identifying a base file stored within the remote storage site.

10. The method of any of claims 2 to 9, further comprising creating a cross-linking between a sub-file and a base file that have been related as a result of the step of determining membership of the file within one of a plurality of groups.

11. The method of any of claims 2 to 10, further comprising determining by comparison of tokens at the remote storage site that a file transmitted to the remote storage site as a sub-file does not have a corresponding base file at the remote storage site and returning a message to the primary storage site to notify the primary storage site that the sub-file will not be stored by the remote storage site.

12. The method of claim 11, further comprising, in response to the receipt of the message, retransmitting a backup of a file of the primary storage site as a base file rather than as a sub-file.

13. A computer program for implementing the method of any preceding claim.

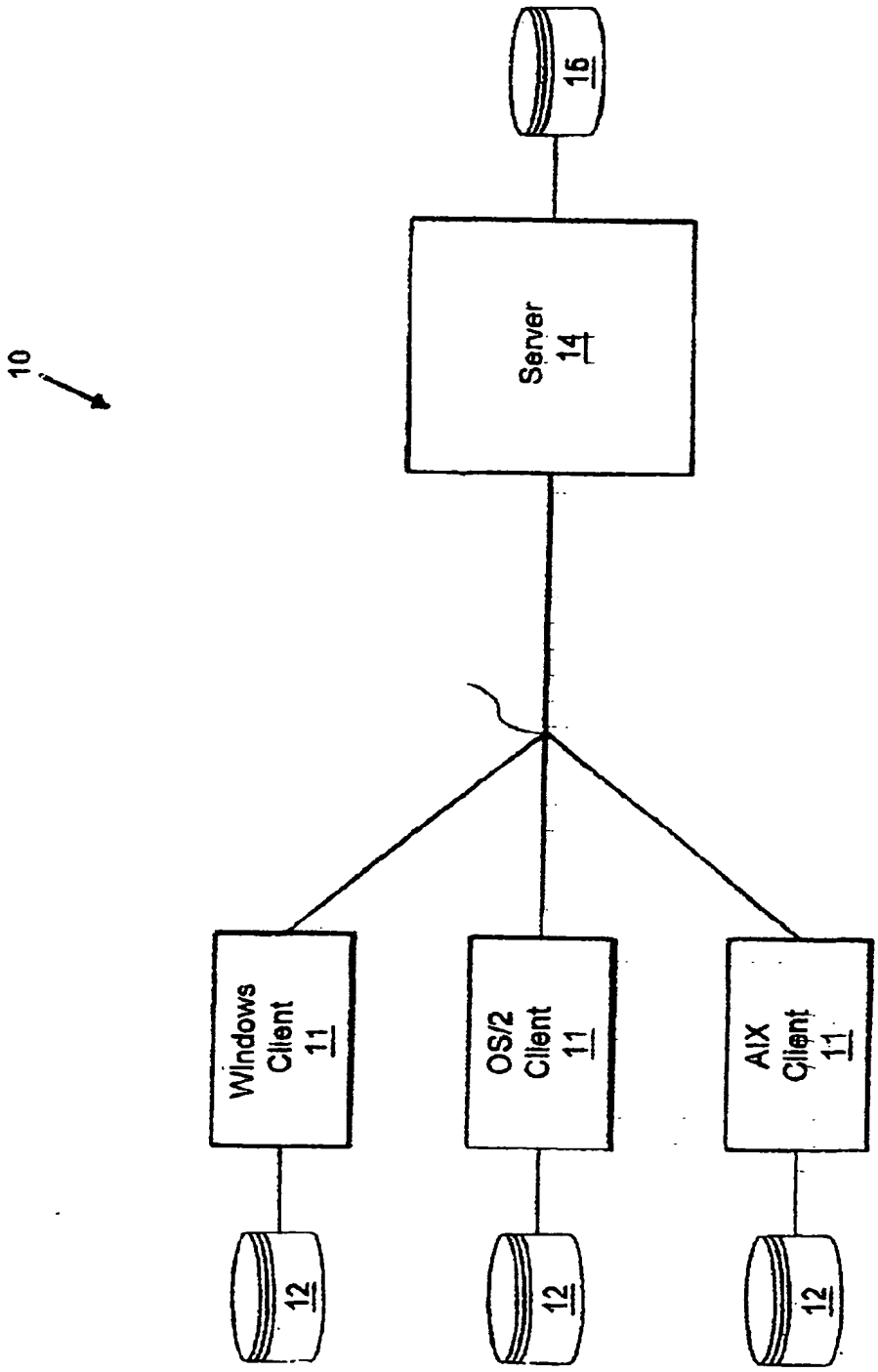
14. A system for relating groups of files in a distributed data storage system having a primary storage site and a remote storage site, the system comprising:

   a token generation module within the primary storage site. the token generation module configured to generate tokens uniquely identifying files transmitted from the primary storage site to the remote storage site;
   a token listing within the remote storage site; and
   a token comparison module within the remote storage site, the token comparison module configured to receive tokens passed in conjunction with transmission of a file from the primary storage site to the remote storage site and to compare the tokens to one or more tokens within

the token listing to establish a relationship of the file with other files previously transmitted from the primary storage site to the remote storage site.

15. The system of claim 14, further comprising a repository within the primary storage site, the repository containing a representation of each of a plurality of base files stored on the remote storage site and also containing a plurality of tokens, each token unique to one of the base files stored on the remote storage site.

16. The system of claim 14 or 15, further comprising a hierarchy of storage devices connected to the remote storage site, a plurality of base files stored within the storage hierarchy, and a plurality of tokens stored within the token listing, each of the plurality of tokens uniquely identifying one of the base files stored within the storage hierarchy.



Prior Art

Fig. 1

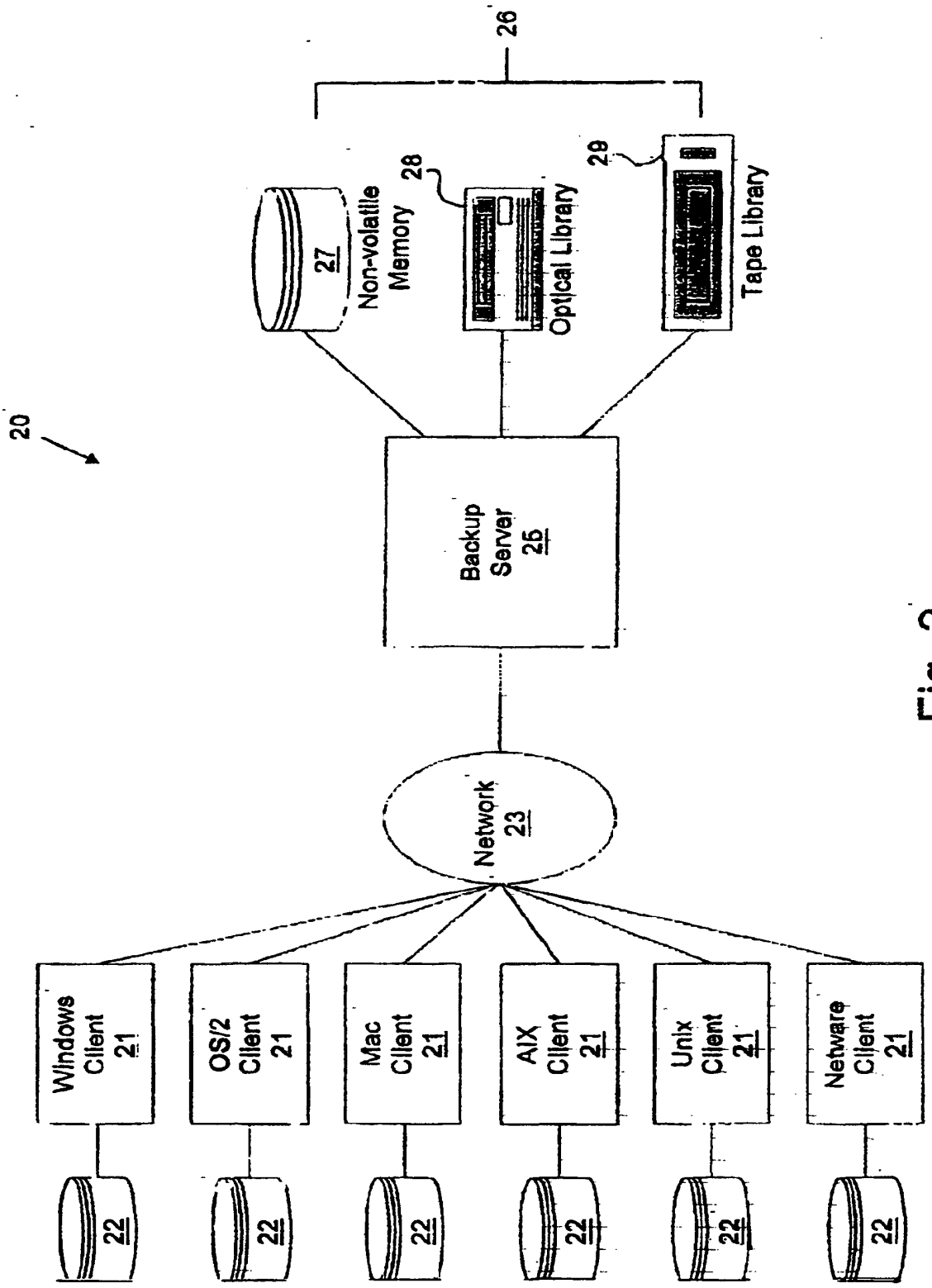


Fig. 2

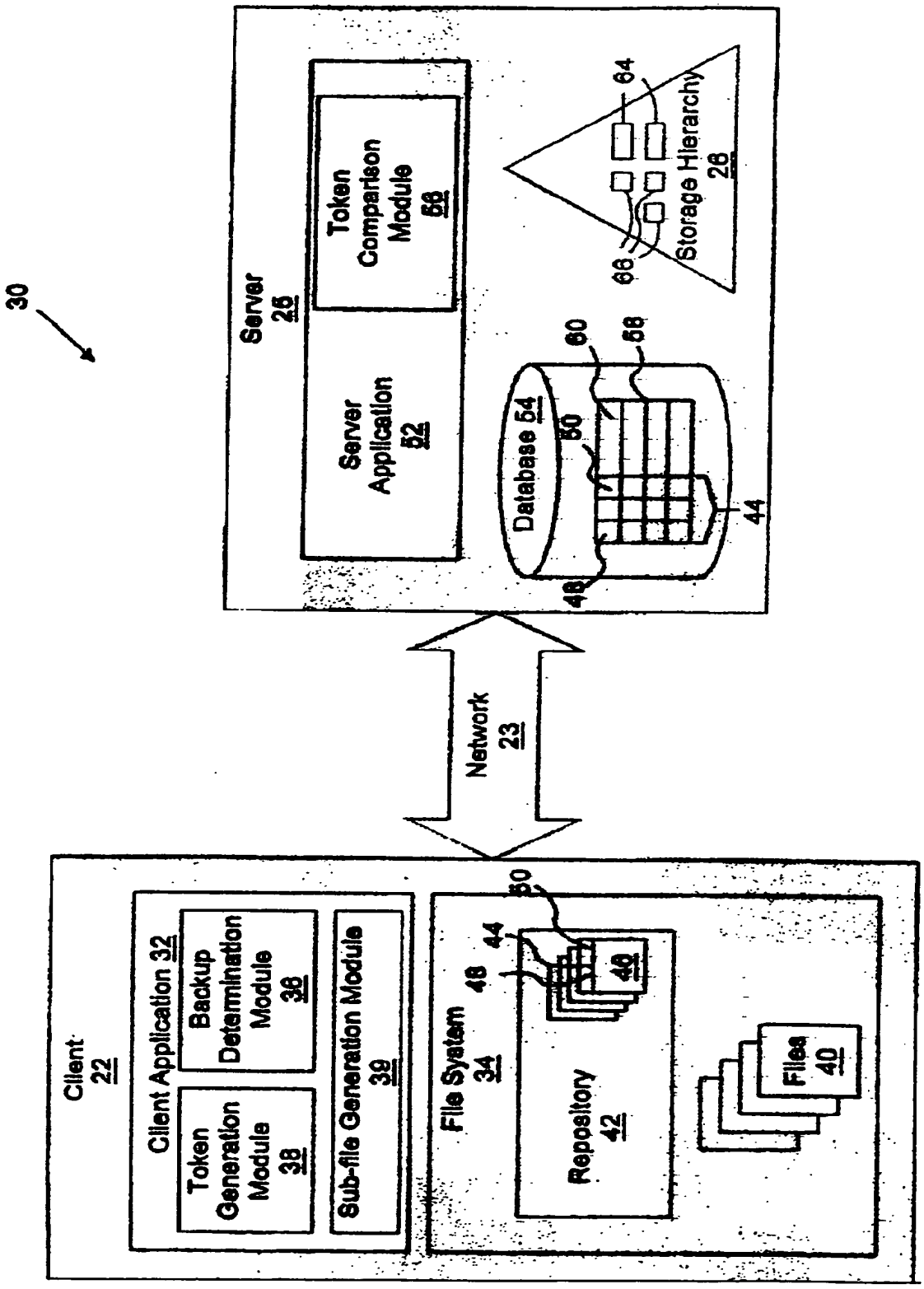


Fig. 3

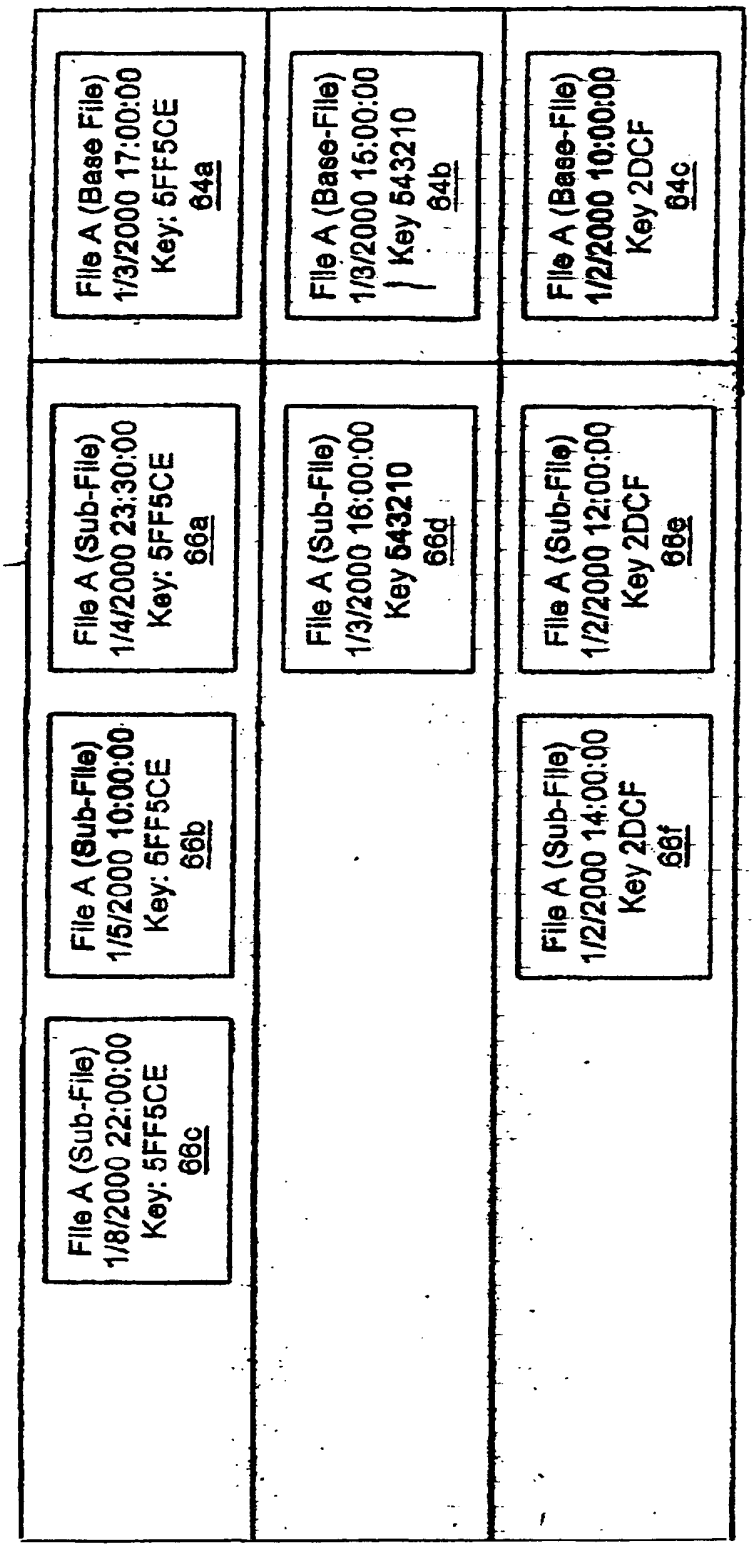


Fig. 4

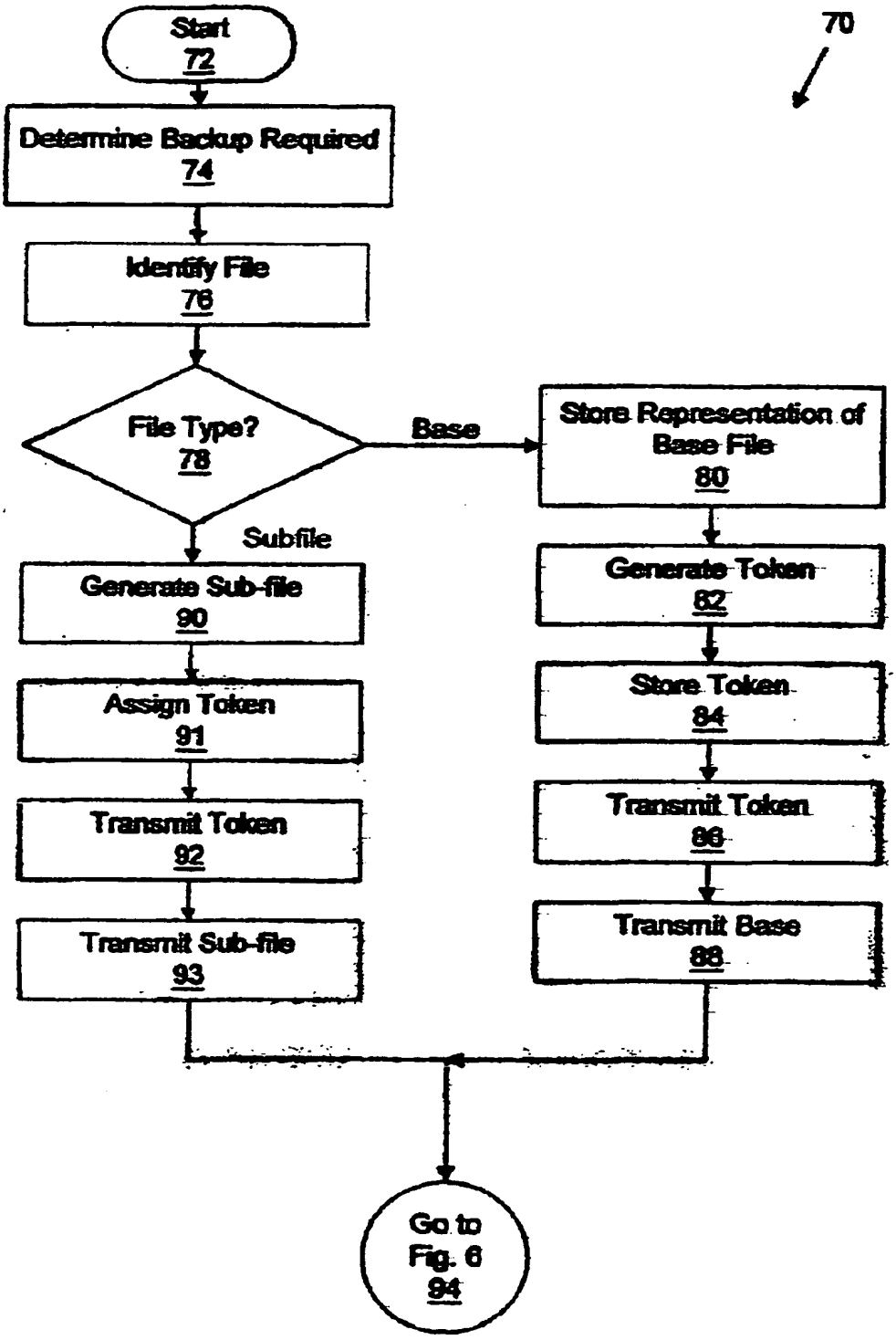


Fig. 5

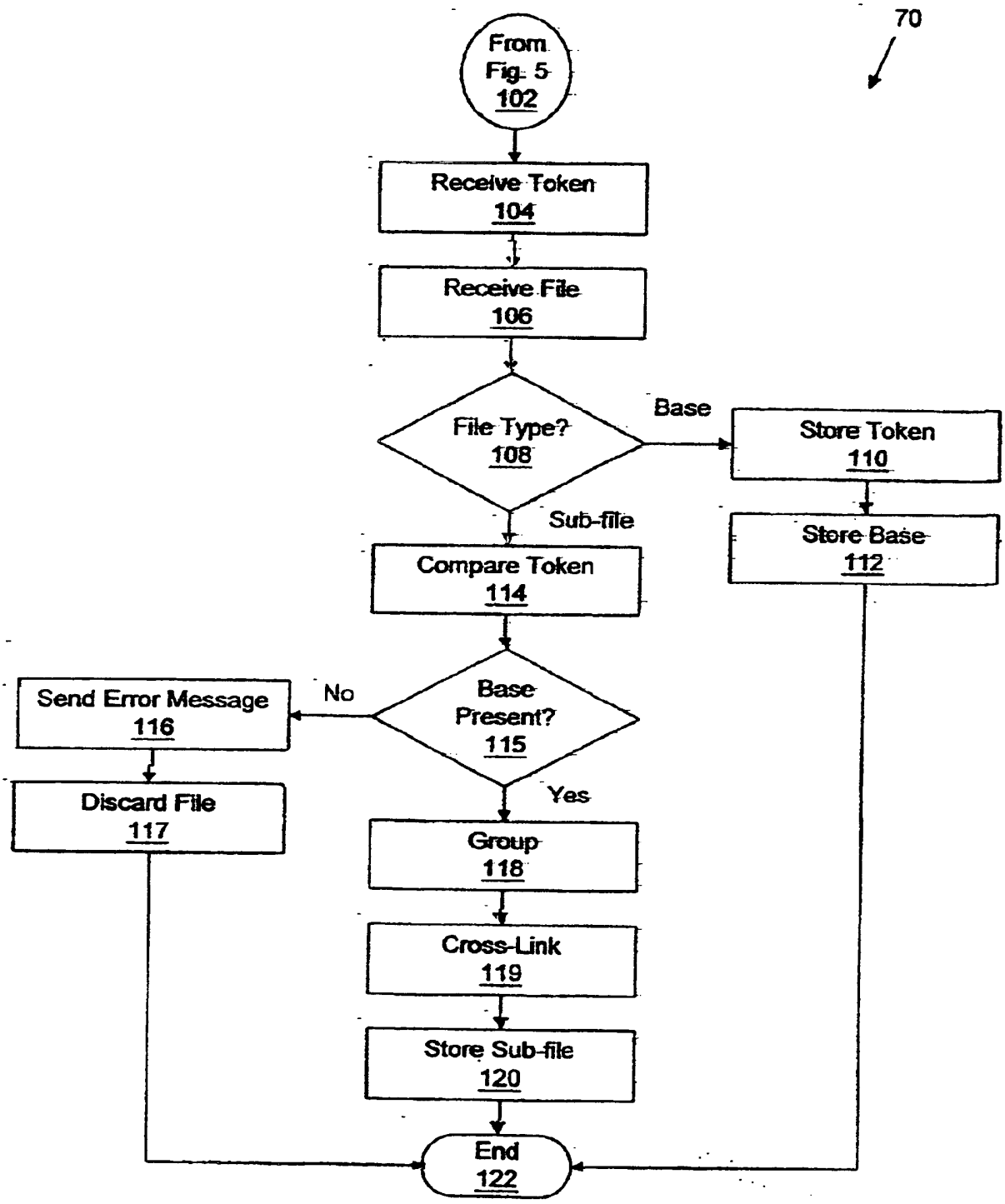


Fig. 6

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 152 352 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
05.10.2005 Bulletin 2005/40

(43) Date of publication A2:
07.11.2001 Bulletin 2001/45

(21) Application number: **01000121.2**

(22) Date of filing: **20.04.2001**

(51) Int Cl.[7]: **G06F 17/30**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **27.04.2000 US 561252**

(71) Applicant: **International Business Machines Corporation**
Armonk, NY 10504 (US)

(72) Inventors:
- **Cannon, David M.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Dawson, Colin S.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Haye, Mark A.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)
- **Smith, James P.**
  Hursley, Winchester, Hampshire SO21 2JN (GB)

(74) Representative: **Burt, Roger James**
IBM United Kingdom Limited,
Intellectual Property Law,
MP 110,
Hursley Park,
Hursley
Winchester, Hampshire SO21 2JN (GB)

(54) System and method for handling files in a distributed data storage environment

(57) A system and method for relating files in a distributed data storage environment is described that allows for positive identification of membership of a file within a group, even in a loosely coupled environment where files are not available for comparison in real time. In disclosed embodiments, base files of a client are stored on a server and are accompanied by tokens uniquely identifying the base files. The tokens are generated on the client and may be derived from the contents of the base file using a digital signature. Each file transmitted to the server is accompanied with a token. Incremental backups may be used, and may employ file differencing. Accordingly, sub-files related to the base files may be transmitted to the server for backup. The sub-files are related to their respective base files using the tokens and are cross-linked to the base files so that any sub-files can be retrieved together with the base file from which the sub-file was derived.

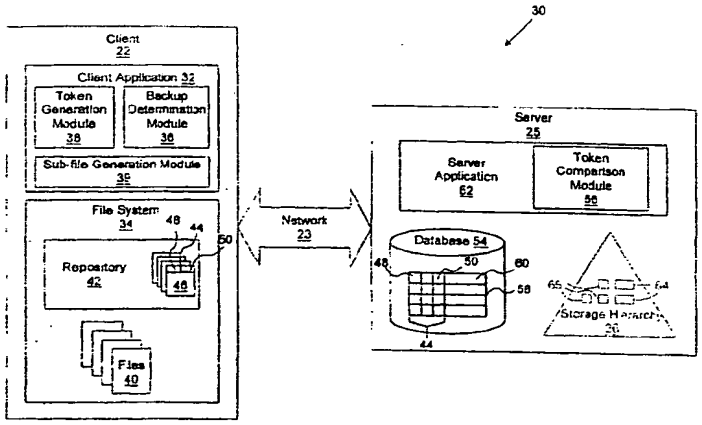


Fig. 3

EP 1 152 352 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 00 0121

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	WO 95/16237 A (INTELLIGENCE QUOTIENT INTERNATIONAL LIMITED) 15 June 1995 (1995-06-15) * page 2, line 9 - page 3, line 9 * * page 5, line 9 - page 8, line 9 * * claims 1-5,12 *	1,13,14	G06F17/30 G06F17/30 G06F11/14
X	US 5 659 614 A (BAILEY, III ET AL) 19 August 1997 (1997-08-19) * abstract * * column 2, line 3 - column 2, line 9 * * column 3, line 2 - column 3, line 13 * * column 5, line 48 - column 10, line 26 * * claims 1,2,28-31 *	1-3, 8-10, 13-16	
A	WO 00/22540 A (SMALL, HUNTER) 20 April 2000 (2000-04-20) * abstract * * page 2, line 4 - page 3, line 7 * * page 5, line 6 - page 7, line 23 *	1-4, 8-10,13, 14	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F
A	"CHANGED DATA ONLY BACKUP AND RECOVERY" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 39, no. 3, March 1996 (1996-03), pages 367-369, XP000581726 ISSN: 0018-8689 * the whole document *	1-4,8,9, 13,14	
A	US 5 634 052 A (MORRIS ET AL) 27 May 1997 (1997-05-27) * abstract * * column 6, line 10 - column 7, line 19 * * column 10, line 1 - column 13, line 18 *	1,2,8, 13,14	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
The Hague	9 August 2005	Abbing, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 00 0121

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

09-08-2005

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9516237 A	15-06-1995	AT 180070 T	15-05-1999
		AU 700681 B2	14-01-1999
		AU 1114695 A	27-06-1995
		CA 2178213 A1	15-06-1995
		DE 69418482 D1	17-06-1999
		DE 69418482 T2	21-10-1999
		DK 733235 T3	15-11-1999
		EP 0733235 A1	25-09-1996
		ES 2131298 T3	16-07-1999
		WO 9516237 A1	15-06-1995
		HK 1014594 A1	05-05-2000
		JP 9506453 T	24-06-1997
		US 5617566 A	01-04-1997
		US 5684991 A	04-11-1997
US 5659614 A	19-08-1997	NONE	
WO 0022540 A	20-04-2000	US 6145012 A	07-11-2000
		AU 1116800 A	01-05-2000
		WO 0022540 A1	20-04-2000
US 5634052 A	27-05-1997	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82